Attribute-Based Encryption for Circuits from Multilinear Maps

Sanjam Garg[1,⋆], Craig Gentry[2,⋆⋆], Shai Halevi[2,⋆⋆], Amit Sahai[1,⋆], and Brent Waters[3,⋆⋆⋆]

[1] UCLA
{sanjamg,sahai}@cs.ucla.edu
[2] IBM Research
{cbgentry,shaih}@us.ibm.com
[3] UT Austin
bwaters@cs.utexas.edu

Abstract. In this work, we provide the first construction of Attribute-Based Encryption (ABE) for general circuits. Our construction is based on the existence of multilinear maps. We prove selective security of our scheme in the standard model under the natural multilinear generalization of the BDDH assumption. Our scheme achieves both Key-Policy and Ciphertext-Policy variants of ABE. Our scheme and its proof of security directly translate to the recent multilinear map framework of Garg, Gentry, and Halevi.

[⋆] Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1228984, 1136174, 1118096, 1065276, 0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

[⋆⋆] This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20202. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

[⋆⋆⋆] Supported by NSF CNS-0915361 and CNS-0952692, CNS-1228599 DARPA via Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship, and Packard Foundation Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

R. Canetti and J.A. Garay (Eds.): CRYPTO 2013, Part II, LNCS 8043, pp. 479–499, 2013.

1 Introduction

In traditional public key encryption a sender will encrypt a message to a targeted individual recipient using the recipient's public key. However, in many applications one may want to have a more general way of expressing who should be able to view encrypted data. Sahai and Waters [SW05] introduced the notion of Attribute-Based Encryption (ABE). There are two variants of ABE: Key-Policy ABE and Ciphertext-Policy ABE [GPSW06]. (We will consider both these variants in this work.) In a Key-Policy ABE system, a ciphertext encrypting a message M is associated with an assignment x of boolean variables. A secret key SK is issued by an authority and is associated with a boolean function f chosen from some class of allowable functions $\mathcal{F}$. A user with a secret key for f can decrypt a ciphertext associated with x, if and only if $f(x) = 1$.

Since the introduction of ABE there have been advances in multiple directions. These include: new proof techniques to achieve adaptive security [LOS$^+$10, OT10, LW12], decentralizing trust among multiple authorities [Cha07, CC09, LW11], and applications to outsourcing computation [PRV12].

However, the central challenge of expanding the *class* of allowable boolean functions $\mathcal{F}$ has been very resistant to attack. Viewed in terms of circuit classes, the work of Goyal *et al* [GPSW06] achieved the best result until now; their construction achieved security essentially for circuits in the complexity class $\mathbf{NC^1}$. This is the class of circuits with depth $\log n$, or equivalently, the class of functions representable by polynomial-size boolean formulas. Achieving ABE for general circuits is arguably the central open direction in this area[1].

Difficulties in Achieving Circuit ABE and the Backtracking Attack. To understand why achieving ABE for general circuits has remained a difficult problem, it is instructive to examine the mechanisms of existing constructions based on bilinear maps. Intuitively, a bilinear map allows one to decrypt using group elements as keys (or key components) as opposed to exponents. By handing out a secret key that consists of group elements, an authority is able to computationally hide some secrets embedded in that key from the key holder herself. In contrast, if a secret key consists of exponents in $\mathbb{Z}_p$ for a prime order group p, as in say an ElGamal type system, then the key holder or collusion of key holders can solve for these secrets using algebra. This computational hiding in bilinear map based systems allows an authority to personalize keys to a user and prevent collusion attacks, which are the central threat.

Using GPSW [GPSW06] as a canonical example we illustrate some of the main principles of decryption. In their system, private keys consist of bilinear group elements for a group of prime order p and are associated with random values $r_y \in \mathbb{Z}_p$ for each leaf node y in the boolean formula f. A ciphertext

[1] We note that if collusions between secret key holders are bounded by a publicly known polynomially-bounded number in advance, then even stronger results are known [SS10, GVW12]. However, throughout this paper we will deal only with the original setting of ABE where unbounded collusions are allowed between adversarial users.

encrypted to descriptor x has randomness $s \in \mathbb{Z}_p$. The decryption algorithm begins by applying a pairing operation to each "satisfied" leaf node and obtains $e(g,g)^{r_y s}$ for each satisfied node y. From this point onward decryption consists solely of finding if there is a linear combination (in the exponent) of the r_y values that can lead to computing $e(g,g)^{\alpha s}$, which will be the "blinding factor" hiding the message M. (The variable $e(g,g)^{\alpha}$ is defined in the public parameters.) The decryption algorithm should be able to find such a linear combination only if $f(x) = 1$. Of particular note is that once the $e(g,g)^{r_y s}$ values are computed the pairing operation plays no further role in decryption. Indeed, it cannot since it is intuitively "used up" on the initial step.

Let's now take a closer look at how GPSW structures a private key for a given boolean formula. Suppose inside a particular boolean formula there exists an OR gate T that received inputs from gates A and B. Then the authority will associate gate T with a value r_T and gates A, B with values $r_A = r_B = r_T$ to match the OR functionality. Now suppose that on a certain input assignment x that gate A evaluates to 1, but gate B evaluates to 0. The decryptor will then learn the "decryption value" $e(g,g)^{sr_A}$ for gate A and can interpolate up by simply by noting that $e(g,g)^{sr_T} = e(g,g)^{sr_A}$. While this structure reflects an OR gate, it also has a critical side effect. The decryption algorithm also learns the decryption value $e(g,g)^{sr_B}$ for gate B *even though gate* B *evaluates to 0* on input x. We call such a discovery a *backtracking attack*.

Boolean formulas are circuits with fanout one. If the fanout is one, then the backtracking attack produces no ill effect since an attacker has nowhere else to go with this information that he has learned. However, suppose we wanted to extend this structure with circuits of fanout of two or more, and that gate B also fed into an AND gate R. In this case the backtracking attack would allow an attacker to act like B was satisfied in the formula even though it was not. This misrepresentation can then be propagated up a different path in the circuit due to the larger fanout. (Interestingly, this form of attack does not involve collusion with a second user.)

We believe that such backtracking attacks are the principle reason that the functionality of existing ABE systems has been limited to circuits of fanout one. Furthermore, we conjecture that since the pairing operation is used up in the initial step, that there is no black-box way of realizing general ABE for circuits from bilinear maps.

Our Results. We present a new methodology for constructing Attribute-Based Encryption systems for circuits of arbitrary fanout. Our method is described using multilinear maps. Cryptography with multilinear maps was first postulated by Boneh and Silverberg [BS02] where they discussed potential applications such as one round, n-way Diffie-Hellman key exchange. However, they also gave evidence that it might be difficult or not possible to find useful multilinear forms within the realm of algebraic geometry. For this reason there has existed a general reluctance among cryptographers to explore multilinear map constructions even though in some constructions such as the Boneh-Goh-Nissim [BGN05] slightly homomorphic encryption system, or the Boneh-Sahai-Waters [BSW06] Traitor

Tracing scheme, there appears to exist direct generalizations of bilinear map solutions.

Very recently, Garg, Gentry, and Halvei [GGH13a] (see [GGH12b] for full version) announced a surprising result. Using ideal lattices they produced a candidate mechanism that would approximate or be the moral equivalent of multilinear maps for many applications. Speculative applications include translations of existing bilinear map constructions and direct generalizations as well as future applications. While the development and cryptanalysis of their tools is at a nascent stage, we believe that their result opens an exciting opportunity to study new constructions using a multilinear map abstraction. The promise of these results is that such constructions can be brought over to their framework or a related future one. We believe that building ABE for circuits is one of the most exciting of these problems due to the challenges discussed above and that existing bilinear map constructions do not have a direct generalization.

Our circuit ABE construction and its proof of security directly translate to the framework of [GGH12b].

We construct an ABE system of the Key-Policy variety where ciphertext descriptors are an n-tuple x of boolean variables and keys are associated with boolean circuits of a max depth ℓ, where both ℓ and n are polynomially bounded and determined at the time of system setup. Our main construction exposition is for circuits that are layered (where gates at depth j get inputs from gates at depth $j-1$) and monotonic (consisting only of AND plus OR gates). Neither one of these impacts our general result as a generic circuit can be transformed into a layered one for the same function with a small amount of overhead. In addition, using De Morgan's law one can build a general circuit from a monotone circuit with negation only appearing at the input wires. We sketch this in Section 2. We finally note that using universal circuits we can realize "Ciphertext-Policy" style ABE systems for circuits.

We use a framework of leveled multilinear maps is that a party can call a group generator $\mathcal{G}(1^\lambda, k)$ to obtain a sequence of groups $\boldsymbol{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ each of large prime[2] order $p > 2^\lambda$ where each comes with a canonical generator $g = g_1, \ldots, g_k$. Slightly abusing notation, if $i + j \leq k$ we can compute a bilinear map operation on $g_i^a \in \mathbb{G}_i, g_j^b \in \mathbb{G}_j$ as $e(g_i^a, g_j^b) = g_{i+j}^{ab}$. These maps can be seen as implementing multilinear maps[3]. It is the need to commit to a certain k value which will require the setup algorithm of our construction to commit to a maximum depth $\ell = k-1$. We will prove security under a generalization of the decision BDH assumption that we call the decision k-multilinear assumption. Roughly, it states that given $g, g^s, g^{c_1}, \ldots, g^{c_k}$ it is hard to distinguish $T = g_k^{s \prod_{j \in [1,k]} c_k}$ from a random element of $\mathbb{G}_k$.

[2] We stress that our techniques do not rely on the groups being of prime order; we only need that certain randomization properties hold in a statistical sense (which hold perfectly over groups of prime order). Therefore, our techniques generalize to other algebraic settings.

[3] We technically consider the existence of a set of bilinear maps $\{e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j} \mid i, j \geq 1;\ i + j \leq k\}$, but will often abuse notation for ease of exposition.

Our Techniques. As discussed there is no apparent generalization of the GPSW methods for achieving ABE for general circuits. We develop new techniques with a focus on preventing the backtracking attacks we described above. Intuitively, we describe our techniques as "move forward and shift"; this *replaces and subsumes* the linear interpolation method of GPSW decryption. In particular, our schemes do not rely on any sophisticated linear secret sharing schemes, as was done by GPSW.

Consider a private key for a given monotonic[4] circuit f with max depth ℓ that works over a group sequence $(\mathbb{G}_1, \ldots, \mathbb{G}_k)$. Each wire w in f is associated by the authority with a random value $r_w \in \mathbb{Z}_p$. A ciphertext for descriptor x will be associated with randomness $s \in \mathbb{Z}_p$. A user should with secret key for f should be able to decrypt if and only if $f(x) = 1$.

The decryption algorithm works by computing $g_{j+1}^{sr_w}$ for each wire w in the circuit that evaluates to 1 on input x. If the wire is 0, the decryptor should not be able to obtain this value. Decryption works from the bottom up. For each input wire w at depth 1, we compute $g_2^{sr_w}$ using a very similar mechanism to GPSW.

We now turn our attention to OR gates to illustrate how we prevent backtracking attacks. Suppose wire w is the output of an OR gate with input wires $A(w), B(w)$ at depth j. Furthermore, suppose on a given input x the wire $A(w)$ evaluates to true and $B(w)$ to false so that the decryptor has $g_j^{sr_{A(w)}}$, but not $g_j^{sr_{B(w)}}$. The private key components associated with wire w are:

$$g^{a_w},\ g^{b_w},\ g_j^{r_w - a_w \cdot r_{A(w)}},\ g_j^{r_w - b_w \cdot r_{B(w)}}$$

for random a_w, b_w. To move decryption onward the algorithm first computes

$$e\left(g^{a_w}, g_j^{sr_{A(w)}}\right) = g_{j+1}^{sa_w r_{A(w)}} .$$

This is the move forward step. Then it computes

$$e\left(g^s, g_j^{r_w - a_w \cdot r_{A(w)}}\right) = g_{j+1}^{s(r_w - a_w r_{A(w)})}.$$

This is the shift step. Multiplying these together gives the desired term $g_{j+1}^{sr_w}$.

Let's examine backtracking attacks in this context. Recall that the attacker's goal is to compute $g_j^{sr_{B(w)}}$ even though wire $B(w)$ is 0, and propagate this forward. From the output term and the fourth key component the attacker can actually inverse the shift process on the B side and obtain $g_{j+1}^{sb_w r_{B(w)}}$. However, since the map e works only in the "forward" direction, it is not possible to invert the move forward step and complete the attack. The crux of our security lies in this idea.

The AND gate mechanism has a similar shift and move forward structure, but requires both inputs for decryption. If this process is applied iteratively to

[4] Recall that assuming that the circuit is monotonic is without loss of generality. Our method also applies to general circuits that involve negations. See Section 2.

an output gate $\tilde{w}$, then one obtains $g_k^{sr_{\tilde{w}}}$. A final header portion of the key and decryption mechanism is used to obtain the message. This portion is similar to prior work.

1.1 Other Related Work

Other recent functionality in a similar vain to ABE includes spatial encryption [Ham11] and regular language functionality [Wat12]. Neither of these seem to point to a path for achieving the general case of circuits. Indeed, [Wat12] argues that backtracking attacks are the reason that the constructions can only support Deterministic Finitie Automata and not Nondeterministic Finite Automata.

An interesting challenge going forward is whether new techniques can be applied to the general case of functional encryption [SW08, BSW11]. In this setting we would like to hide the input x as well as the message. So far the strongest functionality in this setting has been the inner product functionality of Katz, Sahai, and Waters [KSW08] and different variants of this [OT12].

There have been different lattice based constructions of IBE, HIBE, Fuzzy IBE, and ABE [CHKP10, ABB10, ABV$^+$12, Boy13]. While the high level proof structures of these systems follow the earlier bilinear map counterparts closely, the analogies seem to break down at lower level mechanisms. For example, there is more asymmetry in the construction of keys and ciphertexts — in bilinear maps they were both bilinear group elements. Rothblum [Rot12] considers the problem of circular security from bit encryption systems from ℓ-multilinear maps. He considers a different form than us where ℓ group elements of different types are input at once to a multilinear map function. The assumption used is a variant of XDH.

Parno, Raykova and Vaikuntanathan [PRV12] note that delegation from ABE can be achieved from a system that is not collusion resistant, however, they were not able to leverage this to go beyond the boolean formulas of [GPSW06]. The fact that the backtracking attacks described above do not use collusion attacks, but are attacks within a key might help explain this. In our construction the size of group elements and computational cost of group operations grows with the sequence number k and thus the depth of the circuit. Using our system combined with the PRV techniques one can achieve delegated computation where the delegator's work grows only with the depth of the circuit and not the size of the circuit. Since the number of multilinear levels must be bounded at setup, it is not clear if our techniques can be used to improve ABE-type applications in the uniform setting [Wat12].

Concurrent Work. Concurrent to and independent of our work Gorbunov, Vaikuntanathan, and Wee [GVW13] achieve ABE for circuits[5]. One nice feature

[5] Historical note: The present paper which merges [GGH12a] and [SW12] contains only a technical scheme and analysis already present in these works, with some additional elaboration. Thus the scheme and analysis presented here remains independent of [GVW13], and was developed concurrently to it.

of their result is that they reduce security to the Learning with Errors (LWE) problem [Reg05]. Both our result and theirs has "succinct" ciphertexts in that the ciphertext size grows with the maximum depth of the circuits and not the size. Goldwasser, Kalai, Popa, Vaikuntanathan, and Zeldovich [GKP$^+$13] show how to combine such an ABE with fully homomorphic encryption into a succinct single use functional encryption scheme. This in turn implies results for reusable Yao garbled circuits and other applications.

Subsequent Work. Subsequent to our work Garg, Gentry, Sahai, and Waters [GGSW13] showed that a general primitive they termed witness encryption implies circuit ABE if we have witness indistinguishable proofs. Their techniques of moving from witness encryption to ABE are quite different from our direct construction. A drawback of using witness encryption is that current GGSW constructions rely on a different assumption for each NP instance.

1.2 Roadmap

We start by providing preliminary definition in Section 2. We give our construction based on (ideal) multilinear maps in Section 3 which is then translated to the GGH framework [GGH12b] in Section 4. We refer the reader to the full version [GGH$^+$13b] for the proofs of security.

2 Preliminaries

In this section we provide some preliminaries. These include definition of ABE for circuits, discussion of monotone versus general circuits, our multilinear map convention and assumptions, and our circuit notation.

2.1 Definitions for ABE for Circuits

We now give a formal definition of our Attribute-Based Encryption for circuits. Our security definition essentially follows [GPSW06] with the exception that access structures are circuits. Our definition is fit for bounded circuits.

Setup$(1^\lambda, n, \ell)$**.** The setup algorithm takes as input the security parameter, the length n of input descriptors from the ciphertext and a bound ℓ on the circuit depth. It outputs the public parameters PP and a master key MSK.

Encrypt$(\mathrm{PP}, x \in \{0,1\}^n, M)$**.** The encryption algorithm takes as input the public parameters PP, a bit string $x \in \{0,1\}^n$ representing the assignment of boolean variables, and a message m. It outputs a ciphertext CT.

Key Generation$(\mathrm{MSK}, f = (n, q, A, B, \mathtt{GateType}))$**.** The key generation algorithm takes as input the master key MSK and a description of a circuit f, where the depth of f is at most ℓ. The algorithm outputs a private key SK.

Decrypt$(\mathrm{SK}, \mathrm{CT})$**.** The decryption algorithm takes as input a secret key SK and ciphertext CT. The algorithm attempts to decrypt and outputs a message M if successful; otherwise, it outputs a special symbol $\perp$.

Correctness. Consider all messages M, strings $x \in \{0,1\}^n$, and depth ℓ circuits f where $f(x) = 1$. If *Encrypt*$(\mathrm{PP}, x, M) \to \mathrm{CT}$ and *KeyGen*$(\mathrm{MSK}, f) \to \mathrm{SK}$ where PP, MSK were generated from a call to the setup algorithm, then *Decrypt*$(\mathrm{SK}, \mathrm{CT}) = M$.

Security Model for ABE for Circuits. We now briefly describe our security model of *selective security* for ABE for general circuits. We refer the reader to [GGH$^+$13b] for a formal treatment. The selective security definition requires that the attacker first specifies the string x^* and later queries on multiple secret keys, but not ones that can trivially be used to decrypt a ciphertext encrypted under x^*. In particular the adversary can ask secret keys corresponding to any circuit f of his choice, such that $f(x^*) = 0$. The goal of the adversary is then to break semantic security of a challenge ciphertext encrypted under the string x^*.

2.2 General Circuits vs. Monotone Circuits

We begin by observing that there is a folklore transformation that uses De Morgan's rule to transform any general Boolean circuit into an equivalent monotone Boolean circuit, with negation gates only allowed at the inputs. For completeness, we sketch the construction here.

Given a Boolean circuit C, consider the Boolean circuit $\tilde{C}$ that computes the negation of C. Note that such a circuit can be generated by simply recursively applying De Morgan's rule to each gate of C starting at the output gate. The crucial property of this transformation is that in this circuit $\tilde{C}$ each wire computes the negation of the corresponding original wire in C.

Now, we can construct a monotone circuit M by combining C and $\tilde{C}$ as follows: take each negation gate inside C, eliminate it, and replace the output of the negation gate by the corresponding wire in $\tilde{C}$. Do the same for negation gates in $\tilde{C}$, using the wires from C. In the end, this will yield a monotone circuit M with negation gates remaining only at the input level, as desired. The size of M will be no more than twice the original size of C, and the depth of M will be identical to the depth of C, where depth is computed ignoring negation gates. The correctness of this transformation follows trivially from De Morgan's rule.

As a result, we can focus our attention on monotone circuits. Note that inputs to the circuit correspond to boolean variables x_i, and we can simply introduce explicit separate attributes corresponding to $x_i = 0$ and $x_i = 1$. Honest encryptors are instructed to only set one of these two attributes for each variable x_i.

Because of this simple transformation, in the sequel we will only consider ABE for monotone circuits.

2.3 Multilinear Maps

We assume the existence of a group generator $\mathcal{G}$, which takes as input a security parameter n and a positive integer k to indicate the number of allowed pairing operations. $\mathcal{G}(1^\lambda, k)$ outputs a sequence of groups $\mathbb{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ each of large prime order $p > 2^\lambda$. In addition, we let g_i be a canonical generator of $\mathbb{G}_i$ (and is known from the group's description). We let $g = g_1$.

We assume the existence of a set of bilinear maps $\{e_{i,j} : \mathbb{G}_i \times \mathbb{G}_j \to \mathbb{G}_{i+j} \mid i, j \geq 1;\ i + j \leq k\}$. The map $e_{i,j}$ satisfies the following relation:

$$e_{i,j}\left(g_i^a, g_j^b\right) = g_{i+j}^{ab} \ : \ \forall a, b \in \mathbb{Z}_p.$$

We observe that one consequence of this is that $e_{i,j}(g_i, g_j) = g_{i+j}$ for each valid i, j.

When the context is obvious, we will sometimes abuse notation drop the subscripts i, j, For example, we may simply write:

$$e\left(g_i^a, g_j^b\right) = g_{i+j}^{ab}.$$

We define the k-Multilinear Decisional Diffie-Hellman (k-MDDH) assumption as follows:

Assumption 1 (k-Multilinear Decisional Diffie-Hellman: k-MDDH). *The k-Multilinear Decisional Diffie-Hellman (k-MDDH) problem states the following: A challenger runs $\mathcal{G}(1^\lambda, k)$ to generate groups and generators of order p. Then it picks random $s, c_1, \ldots, c_k \in \mathbb{Z}_p$.*

The assumption then states that given $g = g_1, g^s, g^{c_1}, \ldots, g^{c_k}$ it is hard to distinguish $T = g_k^{s \prod_{j \in [1,k]} c_j}$ from a random group element in $\mathbb{G}_k$, with better than negligible advantage (in security parameter λ).

2.4 Circuit Notation

We now define our notation for circuits that adapts the model and notation of Bellare, Hoang, and Rogaway [BHR12] (Section 2.3). For our application we restrict our consideration to certain classes of boolean circuits. First, our circuits will have a single output gate. Next, we will consider layered circuits. In a layered circuit a gate at depth j will receive both of its inputs from wires at depth $j - 1$. Finally, we will restrict ourselves to monotonic circuits where gates are either AND or OR gates of two inputs. [6]

Our circuits will be a five-tuple $f = (n, q, A, B, \texttt{GateType})$. We let n be the number of inputs and q be the number of gates. We define inputs $= \{1, \ldots, n\}$, Wires $= \{1, \ldots, n + q\}$, and Gates $= \{n + 1, \ldots, n + q\}$. The wire $n + q$ is the designated output wire. A : Gates $\to$ Wires/outputwire is a function where $A(w)$ identifies w's first incoming wire and B : Gates $\to$ Wires/outputwire is a function where $B(w)$ identifies w's second incoming wire. Finally, $\texttt{GateType}$: Gates $\to$ $\{\text{AND}, \text{OR}\}$ is a function that identifies a gate as either an AND or OR gate.

We require that $w > B(w) > A(w)$. We also define a function $\texttt{depth}(w)$ where if $w \in$ inputs $\texttt{depth}(w) = 1$ and in general $\texttt{depth}(w)$ of wire w is equal to the shortest path to an input wire plus 1. Since our circuit is layered we require that for all $w \in$ Gates that if $\texttt{depth}(w) = j$ then $\texttt{depth}(A(w)) = \texttt{depth}(B(w)) = j - 1$.

[6] These restrictions are mostly useful for exposition and do not impact functionality. General circuits can be built from non-monotonic circuits. In addition, given a circuit an equivalent layered exists that is larger by at most a polynomial factor.

We will abuse notation and let $f(x)$ be the evaluation of the circuit f on input $x \in \{0,1\}^n$. In addition, we let $f_w(x)$ be the value of wire w of the circuit on input x.

3 Our Construction: Multilinear maps

We now describe our construction. Our main construction is of the Key-Policy form where a key generation algorithm takes in the description of a circuit f and encryption takes in an input x and message M. A user with secret key for f can decrypt if and only if $f(x) = 1$. The system is of the "public index" variety in that only the message M is hidden, while x can be efficiently discovered from the ciphertext, as is standard for ABE. We will also discuss how our KP-ABE scheme yields a Ciphertext-Policy ABE scheme for bounded-size circuits.

The setup algorithm will take as inputs a maximum depth ℓ of all the circuits as well as the input size n for all ciphertexts. All circuits f in our system will be of depth ℓ (have the output gate at depth ℓ) and be layered as discussed in Section 2.4. Using layered circuits and having all circuits be of the same depth is primarily for ease of exposition, as we believe that our construction could directly be adapted to the general case. The fact that setup defines a maximum depth ℓ is more fundamental as the algorithm defines a $k = \ell + 1$ group sequence a k pairings.

We also use the convention here that (multi-bit) messages are be encoded as group elements. In Section 4 we will translate this construction to the GGH setting.

Setup($1^\lambda, n, \ell$). The setup algorithm takes as input a security parameter λ, the maximum depth ℓ of a circuit, and the number of boolean inputs n.

It then runs $\mathcal{G}(1^\lambda, k = \ell + 1)$ that produces groups $\mathbb{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_k)$ of prime order p, with canonical generators $g_1, \ldots, g_k$. We let $g = g_1$. Next, it chooses random $\alpha \in \mathbb{Z}_p$ and $h_1, \ldots, h_n \in \mathbb{G}_1$.

The public parameters, PP, consist of the group sequence description plus:

$$g_k^\alpha, h_1, \ldots, h_n.$$

The master secret key MSK is $(g_{k-1})^\alpha$.

Encrypt(PP, $x \in \{0,1\}^n, M \in \mathbb{G}_k$). The encryption algorithm takes in the public parameters, an descriptor input $x \in \{0,1\}^n$, and a message bit $M \in \mathbb{G}_k$. We use the convention that M is a group element.

The encryption algorithm chooses a random $s \in \mathbb{Z}_p$. It then sets $C_M = M \cdot (g_k^\alpha)^s$. We let S be the set of i such that $x_i = 1$.

The ciphertext is created as

$$\mathrm{CT} = (C_M,\ g^s,\ \forall i \in S\ \ C_i = h_i^s).$$

KeyGen(MSK, $f = (n, q, A, B, \texttt{GateType})$). The algorithm takes in the master secret key and a description f of a circuit. Recall that the circuit has $n + q$ wires with n input wires, q gates and the wire $n + q$ designated as the output wire.

The key generation algorithm chooses random $r_1, \ldots, r_{n+q} \in \mathbb{Z}_p$, where we think of randomness r_w as being associated with wire w. The algorithm produces a “header” component

$$K_H = (g_{k-1})^{\alpha - r_{n+q}}.$$

Next, the algorithm generates key components for every wire w. The structure of the key components depends upon whether w is an input wire, an OR gate, or an AND gate. We describe how it generates components for each case.

– *Input wire*
  By our convention if $w \in [1, n]$ then it corresponds to the w-th input. The key generation algorithm chooses random $z_w \in \mathbb{Z}_p$.
  The key components are:

$$K_{w,1} = g^{r_w} h_w^{z_w},\ K_{w,2} = g^{-z_w}.$$

– *OR gate*
  Suppose that wire $w \in$ Gates and that $\mathtt{GateType}(w) = \text{OR}$. In addition, let $j = \mathtt{depth}(w)$ be the depth of wire w. The algorithm will choose random $a_w, b_w \in \mathbb{Z}_p$. Then the algorithm creates key components:

$$K_{w,1} = g^{a_w},\ K_{w,2} = g^{b_w},\ K_{w,3} = g_j^{r_w - a_w \cdot r_{A(w)}},\ K_{w,4} = g_j^{r_w - b_w \cdot r_{B(w)}}.$$

– *AND gate*
  Suppose that wire $w \in$ Gates and that $\mathtt{GateType}(w) = \text{AND}$. In addition, let $j = \mathtt{depth}(w)$ be the depth of wire w. The algorithm will choose random $a_w, b_w \in \mathbb{Z}_p$. The components are:

$$K_{w,1} = g^{a_w},\ K_{w,2} = g^{b_w},\ K_{w,3} = g_j^{r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}}.$$

We will sometimes refer to the $K_{w,3}, K_{w,4}$ of the AND and OR gates as the “shift” components. This terminology will take on more meaning when we see how they are used during decryption.

The secret key SK output consists of the description of f, the header component K_H and the key components for each wire w.

Decrypt(SK, CT)**.** Suppose that we are evaluating decryption for a secret key associated with a circuit $f = (n, q, A, B, \mathtt{GateType})$ and a cipherext with input x. We will be able to decrypt if $f(x) = 1$.

We begin by observing that the goal of decryption should be to compute $g_k^{\alpha s}$. One can then recover M by computing $M = C_M / g_k^{\alpha s}$. First, there is a header computation where we compute $E' = e(K_H), g^s) = e(g_{k-1}^{\alpha - r_{n+q}}, g^s) = g_k^{\alpha s} g_k^{-r_{n+q} \cdot s}$ Our goal is now reduced to computing $g_k^{r_{n+q} \cdot s}$.

Next, we will evaluate the circuit from the bottom up. Consider wire w at depth j; if $f_w(x) = 1$ then, our algorithm will compute $E_w = (g_{j+1})^{s r_w}$. (If $f_w(x) = 0$ nothing needs to be computed for that wire.) Our decryption algorithm proceeds iteratively starting with computing E_1 and proceeds in order to finally compute E_{n+q}. Computing these values in order ensures that the computation on a depth $j-1$ wire (that evaluates to 1) will be defined before computing for a depth j wire. We show how to compute E_w for all w where $f_w(x) = 1$, again breaking the cases according to whether the wire is an input, AND or OR gate.

- *Input wire*
  By our convention if $w \in [1, n]$ then it corresponds to the w-th input. Suppose that $x_w = f_w(x) = 1$. The algorithm computes:

  $$E_w = e(K_{w,1}, g^s) \cdot e(K_{w,2}, C_w) = e(g^{r_w} h_w^{z_w}, g^s) \cdot e(g^{-z_w}, h_w^s) = g_2^{sr_w}.$$

  We observe that this mechanism is similar to many existing ABE schemes.
- *OR gate*
  Consider a wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{OR}$. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w. Suppose that $f_w(x) = 1$. If $f_{A(w)}(x) = 1$ (the first input evaluated to 1) then we compute:

  $$E_w = e(E_{A(w)}, K_{w,1}) \cdot e(K_{w,3}, g^s) = e(g_j^{sr_{A(w)}}, g^{a_w}) \cdot e(g_j^{r_w - a_w \cdot r_{A(w)}}, g^s) = (g_{j+1})^{sr_w}.$$

  Alternatively, if $f_{A(w)}(x) = 0$, but $f_{B(w)}(x) = 1$, then we compute:

  $$E_w = e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,4}, g^s) = e(g_j^{sr_{B(w)}}, g^{b_w}) \cdot e(g_j^{r_w - b_w \cdot r_{B(w)}}, g^s) = (g_{j+1})^{sr_w}.$$

  Let's examine this mechanism for the case where the first input is 1 ($f_{A(w)}(x) = 1$). In this case the algorithm "moves" the value $E_{A(w)}$ from group $\mathbb{G}_j$ to group $\mathbb{G}_{j+1}$ when pairing it with $K_{w,1}$. It then multiplies it by $e(K_{w,3}, g^s)$ which "shifts" that result to E_w.

  Suppose that $f_{A(w)}(x) = 1$, but $f_{B(w)}(x) = 0$. A critical feature of the mechanism is that an attacker cannot perform a "backtracking" attack to compute $E_{B(w)}$. The reason is that the pairing operation cannot be reverse to go from group $\mathbb{G}_{j+1}$ to group $\mathbb{G}_j$. If this were not the case, it would be debilitating for security as gate $B(w)$ might have fanout greater than 1. This type of backtracking attacking is why existing ABE constructions are limited to circuits with fanout of 1.
- *AND gate*
  Consider a wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{AND}$. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w. Suppose that $f_w(x) = 1$. Then $f_{A(w)}(x) = f_{B(w)}(x) = 1$ and we compute:

  $$E_w = e(E_{A(w)}, K_{w,1}) \cdot e(E_{B(w)}, K_{w,2}) \cdot e(K_{w,3}, g^s)$$

  $$= e(g_j^{sr_{A(w)}}, g^{a_w}) \cdot e(g_j^{sr_{B(w)}}, g^{b_w}) \cdot e(g_j^{r_w - a_w \cdot r_{A(w)} - c_w \cdot r_{B(w)}}, g^s) = (g_{j+1})^{sr_w}.$$

If the $f(x) = f_{n+q}(x) = 1$, then the algorithm will compute $E_{n+q} = g_k^{r_{n+q} \cdot s}$. It finally computes $E' \cdot E_{n+q} = g_k^{\alpha s}$ and tests if this equals C_M, outputting $M = 1$ if so and $M = 0$ otherwise. Correctness holds with high probability.

A Few Remarks. Our OR and AND key components respectively have one and two "shift" components. It is conceivable to have a construction with one shift component for the OR and none for the AND. However, we designed it this way since it made the exposition of our proof provided in the full verison [GGH$^+$13b](in particular the distribution of private keys) easier.

Finally, our construction uses a layered circuit, where a wire at depth j gets its inputs from depth $j' = j - 1$. We could imagine a small modification to our construction which allowed j' to be of any depth less than j. Suppose this were the case for the first input. Then instead of $K_{w,1} = g_1^{a_w}$ we might more generally let $K_{w,1} = (g_{j-j'})^{a_w}$. However, we stick to describing and proving the layered case for simplicity.

4 Our Construction: Based on GGH Graded Algebras

We now describe how to modify our construction to use the GGH [GGH12b] graded algebras analogue of multilinear maps. The translation of our scheme above is straightforward to the GGH setting. We start by providing background on Garg et al.'s lattice-based "approximate" multilinear maps (a.k.a. "graded encoding systems") [GGH12b].

4.1 Graded Encoding Systems: Definition

Garg, Gentry and Halevi (GGH) [GGH12b] defined an "approximate" version of a multilinear group family, which they call a *graded encoding system*. As a starting point, they view g_i^{α} in a multilinear group family as simply an *encoding* of α at "level-i". This encoding permits basic functionalities, such as equality testing (it is easy to check that two level-i encodings encode the same exponent), additive homomorphism (via the group operation in $\mathbb{G}_i$), and bounded multiplicative homomorphism (via the multilinear map e). They retain the notion of a somewhat homomorphic encoding with equality testing, but they use probabilistic encodings, and replace the multilinear group family with "less structured" sets of encodings related to lattices.

Abstractly, their n-graded encoding system for a ring R includes a system of sets $\mathcal{S} = \{S_i^{(\alpha)} \subset \{0,1\}^* : i \in [0,n], \alpha \in R\}$ such that, for every fixed $i \in [0,n]$, the sets $\{S_i^{(\alpha)} : \alpha \in R\}$ are disjoint (and thus form a partition of $S_i \overset{\text{def}}{=} \bigcup_\alpha S_i^{(\alpha)}$). The set $S_i^{(\alpha)}$ consists of the "level-i encodings of α". Moreover, the system comes equipped with efficient procedures, as follows:[7]

[7] Since GGH's realization of a graded encoding system uses "noisy" encodings over ideal lattices, the procedures incorporate information about the magnitude of the noise.

Instance Generation. The randomized $\mathsf{InstGen}(1^\lambda, 1^n)$ takes as input the security parameter λ and integer n. The procedure outputs $(\mathsf{params}, \mathbf{p}_{zt})$, where params is a description of an n-graded encoding system as above, and $\mathbf{p}_{zt}$ is a level-n "zero-test parameter".

Ring Sampler. The randomized $\mathsf{samp}(\mathsf{params})$ outputs a "level-zero encoding" $a \in S_0$, such that the induced distribution on α such that $a \in S_0^{(\alpha)}$ is statistically uniform.

Encoding. The (possibly randomized) $\mathsf{enc}(\mathsf{params}, i, a)$ takes $i \in [n]$ and a level-zero encoding $a \in S_0^{(\alpha)}$ for some $\alpha \in R$, and outputs a level-i encoding $u \in S_i^{(\alpha)}$ for the same α.

Re-Randomization. The randomized $\mathsf{reRand}(\mathsf{params}, i, u)$ re-randomizes encodings to the same level, as long as the initial encoding is under a given noise bound. Specifically, for a level $i \in [n]$ and encoding $u \in S_i^{(\alpha)}$, it outputs another encoding $u' \in S_i^{(\alpha)}$. Moreover for any two encodings $u_1, u_2 \in S_i^{(\alpha)}$ whose noise bound is at most some b, the output distributions of $\mathsf{reRand}(\mathsf{params}, i, u_1)$ and $\mathsf{reRand}(\mathsf{params}, i, u_2)$ are statistically the same.

Addition and negation. Given params and two encodings at the same level, $u_1 \in S_i^{(\alpha_1)}$ and $u_2 \in S_i^{(\alpha_2)}$, we have $\mathsf{add}(\mathsf{params}, u_1, u_2) \in S_i^{(\alpha_1+\alpha_2)}$, and $\mathsf{neg}(\mathsf{params}, u_1) \in S_i^{(-\alpha_1)}$, subject to bounds on the noise.

Multiplication. For $u_1 \in S_{i_1}^{(\alpha_1)}$, $u_2 \in S_{i_2}^{(\alpha_2)}$, we have $\mathsf{mult}(\mathsf{params}, u_1, u_2) \in S_{i_1+i_2}^{(\alpha_1 \cdot \alpha_2)}$.

Zero-test. The procedure $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u)$ outputs 1 if $u \in S_n^{(0)}$ and 0 otherwise. Note that in conjunction with the procedure for subtracting encodings, this gives us an equality test.

Extraction. This procedure extracts a "canonical" and "random" representation of ring elements from their level-n encoding. Namely $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u)$ outputs (say) $K \in \{0,1\}^\lambda$, such that:

(a) With overwhelming probability over the choice of $\alpha \in R$, for any two $u_1, u_2 \in S_n^{(\alpha)}$, $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_1) = \mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u_2)$,

(b) The distribution $\{\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u) : \alpha \in R, u \in S_n^{(\alpha)}\}$ is statistically uniform over $\{0,1\}^\lambda$.

We can extend add and mult to handle more than two encodings as inputs, by applying the binary versions of add and mult iteratively. Also, we use the canonicalizing encoding algorithm (as defined in Remark 2 of [GGH12b]) $\mathsf{cenc}_\ell(\mathsf{params}, i, a)$ which takes as input encoding of a and generates another encoding according to a "nice" distribution. This parameter ℓ essentially captures the noise present in the encodings. In our scheme the maximum value ℓ takes will be a small constant.

Recall that the k-multilinear assumption for the graded encodings as follows:

Assumption 2 (k-GMDDH Assumption). *The k-Graded Multilinear Decisional Diffie-Hellman (k-GMDDH) assumption states the following: Given* $\mathsf{cenc}_1($ $\mathsf{params}, 1, s)$, $\mathsf{cenc}_1(\mathsf{params}, 1, c_1), \ldots, \mathsf{cenc}_1(\mathsf{params}, 1, c_k)$*, it is hard to distinguish*

$T = \mathsf{cenc}_1(\mathsf{params}, k, s\prod_{j\in[1,k]} c_j)$ *from* $T = \mathsf{cenc}_1(\mathsf{params}, k, \mathsf{samp}(\mathsf{params}))$, *with better than negligible advantage (in security parameter* λ*), where* $(\mathsf{params}, \mathbf{p}_{zt}$ $) \leftarrow \mathsf{InstGen}(1^\lambda, 1^k)$. *and* $s, c_1, \ldots, c_k \leftarrow \mathsf{samp}(\mathsf{params})$.

4.2 Graded Encoding Systems: Realization

Concretely, GGH's n-graded encoding system works as follows. (This is a whirlwind overview; see [GGH12b] for details.) The system uses three rings. First, it uses the ring of integers $\mathcal{O}$ of the m-th cyclotomic field. This ring is typically represented as the ring of polynomials $\mathcal{O} = \mathbb{Z}[x]/(\Phi_m(x))$, where $\Phi_m(x)$ is the m-th cyclotomic polynomial, which has degree $N = \phi(m)$. Second, for some suitable integer modulus q, it uses the quotient ring $\mathcal{O}/(q) = \mathbb{Z}_q[x]/(\Phi_m(x))$, similar to the NTRU encryption scheme [HPS98]. The encodings live in $\mathcal{O}/(q)$. Finally, it uses the quotient ring $R = \mathcal{O}/\mathcal{I}$, where $\mathcal{I} = \langle g\rangle$ is a principal ideal of $\mathcal{O}$ that is generated by g and where $|\mathcal{O}/\mathcal{I}|$ is a large prime. This is the ring "R" referred to above; elements of R are what is encoded.

What does a GGH encoding look like? For a fixed random $z \in \mathcal{O}/(q)$, an element of $S_i^{(\alpha)}$ – that is, a level-i encoding of $\alpha \in R$ – has the form $e/z^i \in \mathcal{O}/(q)$, where $e \in \mathcal{O}$ is a "small" representative of the coset $\alpha + \mathcal{I}$ (it has coefficients that are very small compared to q). To add encodings $e_1/z^i \in S_i^{(\alpha_1)}$ and $e_2/z^i \in S_i^{(\alpha_2)}$, just add them in $\mathcal{O}/(q)$ to obtain $(e_1 + e_2)/z^i$, which is in $S_i^{(\alpha_1+\alpha_2)}$ if $e_1 + e_2$ is "small". To mult encodings $e_1/z^{i_1} \in S_{i_1}^{(\alpha_1)}$ and $e_2/z^{i_2} \in S_{i_2}^{(\alpha_2)}$, just multiply them in $\mathcal{O}/(q)$ to obtain $e_1 \cdot e_2/z^{i_1+i_2}$, which is in $S_{i_1+i_2}^{(\alpha_1\cdot\alpha_2)}$ if $e_1 \cdot e_2$ is "small". This smallness condition limits the GGH encoding system to degree polynomial in the security parameter. Intuitively, dividing encodings does not "work", since the resulting denominator has a nontrivial term that is not z.

The GGH params allow everyone to generate encodings of random (known) values. The params include a level-1 encoding of 1 (from which one can generate encodings of 1 at other levels), and (for each $i \in [n]$) a sufficient number of level-i encodings of 0 to enable re-randomization. To encode (say at level-1), run $\mathsf{samp}(\mathsf{params})$ to sample a small element a from $\mathcal{O}$, e.g. according to a discrete Gaussian distribution. For a Gaussian with appropriate deviation, this will induce a statistically uniform distribution over the cosets of $\mathcal{I}$. Then, multiply a with the level-1 encoding of 1 to get a level-1 encoding u of $a \in R$. Finally, run $\mathsf{reRand}(\mathsf{params}, 1, u)$, which involves adding a random Gaussian linear combination of the level-1 encodings of 0, whose noisiness (i.e., numerator size) "drowns out" the initial encoding. The parameters for the GGH scheme can be instantiated such that the re-randomization procedure can be used for any pre-specified polynomial number of times.

To permit testing of whether a level-n encoding $u = e/z^n \in S_n$ encodes 0, GGH publishes a level-n zero-test parameter $\mathbf{p}_{zt} = hz^n/g$, where h is "somewhat small"[8] and g is the generator of $\mathcal{I}$. The procedure $\mathsf{isZero}(\mathsf{params}, \mathbf{p}_{zt}, u)$ simply

[8] Its coefficients are on the order of (say) $q^{2/3}$, while other terms – such as a numerator e or the principal ideal generator g – are much, much smaller.

computes $\mathbf{p}_{zt} \cdot u$ and tests whether its coefficients are small modulo q. If u encodes 0, then $e \in \mathcal{I}$ and equals $g \cdot c$ for some (small) c, and thus $\mathbf{p}_{zt} \cdot u = h \cdot c$ has no denominator and is small modulo q. If u encodes something nonzero, $\mathbf{p}_{zt} \cdot u$ has g in the denominator and is not small modulo q. The $\mathsf{ext}(\mathsf{params}, \mathbf{p}_{zt}, u)$ procedure works by applying a strong extractor to the most significant bits of $\mathbf{p}_{zt} \cdot u$. For any two $u_1, u_2 \in S_n^{(\alpha)}$, we have (subject to noise issues) $u_1 - u_2 \in S_n^{(0)}$, which implies $\mathbf{p}_{zt}(u_1 - u_2)$ is small, and hence $\mathbf{p}_{zt} \cdot u_1$ and $\mathbf{p}_{zt} \cdot u_2$ have the same most significant bits (for an overwhelming fraction of α's).

4.3 Our Construction

Now we provide our construction in GGH's n-graded encoding system. **For ease of notation on the reader, we suppress repeated params arguments that are provided to every algorithm.**. Thus, for instance, we will write $\alpha \leftarrow \mathsf{samp}()$ instead of $\alpha \leftarrow \mathsf{samp}(\mathsf{params})$. Note that in our scheme, there will only ever be a single uniquely chosen value for params throughout the scheme, so there is no cause for confusion.

Setup$(1^\lambda, n, \ell)$**.** The setup algorithm takes as input, a security parameter λ, the maximum depth ℓ of a circuit, and the number of boolean inputs n.

It then runs $(\mathbf{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^{k=\ell+1})$. Recall that params will be implicitly given as input to all GGH-related algorithms below. Next, it samples $\alpha, \hat{h}_1, \ldots, \hat{h}_n \leftarrow \mathsf{samp}()$.

The public parameters, PP, consist of $\mathbf{p}_{zt}$, plus:

$$H = \mathsf{cenc}_2(k, \alpha), h_1 = \mathsf{cenc}_2(1, \hat{h}_1), \ldots, h_n = \mathsf{cenc}_2(1, \hat{h}_n).$$

The master secret key MSK is α.

Encrypt$(\mathrm{PP}, x \in \{0,1\}^n, M \in \{0,1\})$**.** The encryption algorithm takes in the public parameters, an descriptor input $x \in \{0,1\}^n$, and a message bit $M \in \{0,1\}$.

The encryption algorithm chooses a random $s \leftarrow \mathsf{samp}()$. If $M = 0$ it sets C_M to be a random value:

$$C_M = \mathsf{cenc}_3(k, \mathsf{samp}())$$

otherwise it lets

$$C_M = \mathsf{cenc}_3(k, H \cdot s).$$

Next, let S be the set of i such that $x_i = 1$.

The ciphertext is created as

$$\mathrm{CT} = (C_M,\ \tilde{s} = \mathsf{cenc}_1(1, s),\ \forall i \in S\ \ C_i = \mathsf{cenc}_3(1, h_i \cdot s)).$$

KeyGen$(\mathrm{MSK} = \alpha, f = (n, q, A, B, \mathtt{GateType}))$**.** The algorithm takes in the master secret key and a description f of a circuit. Recall, that the circuit has $n + q$ wires with n input wires, q gates and the wire $n + q$ designated as the output wire.

The key generation algorithm chooses random $r_1, \ldots, r_{n+q} \leftarrow \mathsf{samp}()$, where we think of randomness r_w as being associated with wire w. The algorithm produces a "header" component

$$K_H = \mathsf{cenc}_3(k-1, \alpha - r_{n+q}).$$

Next, the algorithm generates key components for every wire w. The structure of the key components depends upon if w is an input wire, an OR gate, or an AND gate. We describe how it generates components for each case.

- *Input wire*
  By our convention if $w \in [1, n]$ then it corresponds to the w-th input. The key generation algorithm chooses random $z_w \leftarrow \mathsf{samp}()$.
  The key components are:

$$K_{w,1} = \mathsf{cenc}_3(1, \mathsf{enc}(1, r_w) + h_w \cdot z_w),\ K_{w,2} = \mathsf{cenc}_3(1, -z_w).$$

- *OR gate*
  Suppose that wire $w \in$ Gates and that $\texttt{GateType}(w) =$ OR. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w. The algorithm will choose random $a_w, b_w \leftarrow \mathsf{samp}()$. Then the algorithm creates key components:

$$K_{w,1} = \mathsf{cenc}_3(1, a_w),\ K_{w,2} = \mathsf{cenc}_3(1, b_w),$$

$$K_{w,3} = \mathsf{cenc}_3(j, r_w - a_w \cdot r_{A(w)}),\ K_{w,4} = \mathsf{cenc}_3(j, r_w - b_w \cdot r_{B(w)}).$$

- *AND gate*
  Suppose that wire $w \in$ Gates and that $\texttt{GateType}(w) =$ AND. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w. The algorithm will choose random $a_w, b_w \leftarrow \mathsf{samp}()$.

$$K_{w,1} = \mathsf{cenc}_3(1, a_w),\ K_{w,2} = \mathsf{cenc}_3(1, b_w),$$

$$K_{w,3} = \mathsf{cenc}_3(j, r_w - a_w \cdot r_{A(w)} - b_w \cdot r_{B(w)}).$$

We will sometimes refer to the $K_{w,3}, K_{w,4}$ of the AND and OR gates as the "shift" components. This terminology will take on more meaning when we see how they are used during decryption.

The secret key SK output consists of the description of f, the header component K_H and the key components for each wire w.

Decrypt(SK, CT**).** Suppose that we are evaluating decryption for a secret key associated with a circuit $f = (n, q, A, B, \texttt{GateType})$ and a cipherext with input x. We will be able to decrypt if $f(x) = 1$.

We begin by observing that the goal of decryption should be to compute a level k encoding of $\alpha \cdot s$ such that we can test if this is equal to C_M. First, there is a header computation where we compute $E' = K_H \cdot \tilde{s}$. Note that E' should thus be a level k encoding of $\alpha s - r_{n+q} \cdot s$. Our goal is now reduced to computing a level k encoding of $r_{n+q} \cdot s$.

Next, we will evaluate the circuit from the bottom up. Consider wire w at depth j; if $f_w(x) = 1$ then, our algorithm will compute E_w to be a level $j+1$ encoding of sr_w. Note that if $f_w(x) = 0$ nothing needs to be computed for that wire, since we have a monotonic circuit. Our decryption algorithm proceeds iteratively starting with computing E_1 and proceeds in order to finally compute E_{n+q}. Computing these values in order ensures that the computation on a depth $j-1$ wire (that evaluates to 1) will be defined before computing for a depth j wire. We show how to compute E_w for all w where $f_w(x) = 1$, again breaking the cases according to whether the wire is an input, AND or OR gate.

- *Input wire*
  By our convention if $w \in [1, n]$ then it corresponds to the w-th input. Suppose that $x_w = f_w(x) = 1$. The algorithm computes:
  $$E_w = K_{w,1} \cdot \tilde{s} + K_{w,2} \cdot C_w.$$
  Thus, E_w computes a level 2 encoding of $(r_w + \hat{h_w} \cdot z_w) \cdot s + (-z_w) \cdot \hat{h_w} \cdot s = sr_w$.
- *OR gate*
  Consider a wire $w \in$ Gates and that $\texttt{GateType}(w) = \text{OR}$. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w. Suppose that $f_w(x) = 1$. If $f_{A(w)}(x) = 1$ (the first input evaluated to 1) then we compute:
  $$E_w = E_{A(w)} \cdot K_{w,1} + K_{w,3} \cdot \tilde{s}.$$
  Thus, E_w computes a level $j+1$ encoding of $sr_{A(w)} \cdot a_w + (r_w - a_w \cdot r_{A(w)}) \cdot s = sr_w$.

  Alternatively, if $f_{A(w)}(x) = 0$, but $f_{B(w)}(x) = 1$, then we compute:
  $$E_w = E_{B(w)} \cdot K_{w,2} + K_{w,4} \cdot \tilde{s}.$$
  This similarly computes a level $j+1$ encoding of $sr_{B(w)} \cdot b_w + (r_w - b_w \cdot r_{B(w)}) \cdot s = sr_w$.

  Let's examine this mechanism for the case where the first input is 1 ($f_{A(w)}(x) = 1$). In this case the algorithm "moves" the value $E_{A(w)}$ from level j to level $j+1$ when multiplying it with $K_{w,1}$. It then adds it to $K_{w,3} \cdot \tilde{s}$ which "shifts" that result to E_w.

  Suppose that $f_{A(w)}(x) = 1$, but $f_{B(w)}(x) = 0$. A critical feature of the mechanism is that an attacker cannot perform a "backtracking" attack to compute $E_{B(w)}$. The reason is that the GGH encoding cannot be reversed to go from level $j+1$ to level j. (See [GGH12b] for details on why this is the case.) If this were not the case, it would be debilitating for security as gate $B(w)$ might have fanout greater than 1. This type of backtracking attacking is why existing ABE constructions are limited to circuits with fanout of 1.

- *AND gate*
  Consider a wire $w \in$ Gates and that $\texttt{GateType}(w) =$ AND. In addition, let $j = \texttt{depth}(w)$ be the depth of wire w. Suppose that $f_w(x) = 1$. Then $f_{A(w)}(x) = f_{B(w)}(x) = 1$ and we compute:

  $$E_w = E_{A(w)} \cdot K_{w,1} + E_{B(w)} \cdot K_{w,2} + K_{w,3} \cdot \tilde{s}.$$

  Note that this computes a level $j+1$ encoding of sr_w in a manner analogous to above.

If $f(x) = f_{n+q}(x) = 1$, then the algorithm will compute E_{n+q} to be a level k encoding of $r_{n+q} \cdot s$. It finally computes $E' + E_{n+q}$ which is a level k encoding of αs and tests if this equals C_M using $\mathsf{isZero}(\mathbf{p}_{zt}, E' + E_{n+q} - C_M)$, outputting $M = 1$ if so and $M = 0$ otherwise. Correctness holds with high probability.

A Quick Remark about Message Length. Our encryption algorithm takes as input a single bit message. We can extend this to longer messages using the ext algorithm provided by the GGH encoding (see Section 4.1). We restrict ourselves to single bit messages for clarity of the scheme and proof of security. We postpone the proof itself to the full version [GGH$^+$13b].

References

[ABB10] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)

[ABV$^+$12] Agrawal, S., Boyen, X., Vaikuntanathan, V., Voulgaris, P., Wee, H.: Functional encryption for threshold functions (or fuzzy IBE) from lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 280–297. Springer, Heidelberg (2012)

[BGN05] Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)

[BHR12] Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. Cryptology ePrint Archive, Report 2012/265 (2012), http://eprint.iacr.org/

[Boy13] Boyen, X.: Attribute-based functional encryption on lattices. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 122–142. Springer, Heidelberg (2013)

[BS02] Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. IACR Cryptology ePrint Archive 2002:80 (2002)

[BSW06] Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)

[BSW11] Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)

[CC09] Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: ACM Conference on Computer and Communications Security, pp. 121–130 (2009)

[Cha07] Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)

[CHKP10] Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)

[GGH12a] Garg, S., Gentry, C., Halevi, S.: Attribute-based encryption for circuits from multilinear maps (2012) (manuscript)

[GGH12b] Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices and applications. IACR Cryptology ePrint Archive, 2012:610 (2012)

[GGH13a] Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013)

[GGH$^+$13b] Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. Cryptology ePrint Archive, Report 2013/128 (2013), http://eprint.iacr.org/

[GGSW13] Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: STOC, pp. 467–476 (2013)

[GKP$^+$13] Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Succinct functional encryption and applications: Reusable garbled circuits and beyond. In: STOC, pp. 555–564 (2013)

[GPSW06] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)

[GVW12] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012)

[GVW13] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: STOC, pp. 545–554 (2013)

[Ham11] Hamburg, M.: Spatial encryption. IACR Cryptology ePrint Archive, 2011:389 (2011)

[HPS98] Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)

[KSW08] Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)

[LOS$^+$10] Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (Hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)

[LW11] Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011)

[LW12] Lewko, A., Waters, B.: New proof methods for attribute-based encryption: Achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012)

[OT10] Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)

[OT12] Okamoto, T., Takashima, K.: Adaptively attribute-hiding (Hierarchical) inner product encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012)

[PRV12] Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012)

[Reg05] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC, pp. 84–93 (2005)

[Rot12] Rothblum, R.: On the circular security of bit-encryption. Cryptology ePrint Archive, Report 2012/102 (2012), http://eprint.iacr.org/

[SS10] Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: ACM Conference on Computer and Communications Security, pp. 463–472 (2010)

[SW05] Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)

[SW08] Sahai, A., Waters, B.: Slides on functional encryption. PowerPoint presentation (2008), http://www.cs.utexas.edu/ bwaters/presentations/files/functional.ppt

[SW12] Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. IACR Cryptology ePrint Archive 2012:592 (2012)

[Wat12] Waters, B.: Functional encryption for regular languages. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 218–235. Springer, Heidelberg (2012)